

# 异构环境下非均质性最小冗余分布式云存储系统

胡力文<sup>1</sup>, 李启明<sup>2</sup>

(1 江苏财会职业学院 信息系, 江苏 连云港 222061;

2 上海海事大学 信息工程学院, 上海 201306)

**摘 要:** 为提高分布式云存储系统构建的合理性, 提出一种异构环境下考虑非均质性的最小冗余分布式云存储系统构建方法. 首先, 开发了异构环境中, 进行数据可用性计算的分析框架. 由于在存储节点增长时数据可用性计算非常复杂, 提出了一种以较少计算代价的方法来估计真实值的蒙特卡洛方法. 其次, 由于决定在每个主机上分配的最佳冗余量是很难计算的, 提出了一种基于粒子群算法(PSO)的分配方法. 最后, 提供了一个简单的迭代算法来确定保证不同存储应用程序的数据可用性所需的最小冗余. 实验结果显示, 所提方法可实现数据冗余和存储成本的降低.

**关键词:** 异构环境; 非均质性; 最小冗余; 分布式; 云存储

**中图分类号:** TP391

**文献标识码:** A

**文章编号:** 1000-7180(2018)05-0021-06

## Minimum Redundancy Distributed Cloud Storage System Considering Heterogeneity in Heterogeneous Environment

HU Li-wen<sup>1</sup>, LI Qi-ming<sup>2</sup>

(1 Information Department, Jiangsu College of Finance & Accounting, Lianyungang 222061, China;

2 Information Engineering College, Shanghai Maritime University, Shanghai 201306, China)

**Abstract:** In order to improve the rationality of distributed cloud storage system construction, a minimum redundant distributed cloud storage system construction method under heterogeneous environment is proposed. Firstly, an analytical framework for computing data availability in heterogeneous environments is developed. Because the computation of data availability is very complicated when the storage nodes are growing, a Monte Carlo method is proposed to estimate the real values with less computational cost. Secondly, since it is difficult to compute the optimal redundancy allocation for each host, a method based on particle swarm optimization (PSO) is proposed. Finally, a simple iterative algorithm is provided to determine the minimum redundancy required to ensure data availability for different storage applications. Experimental results show that the proposed method can reduce data redundancy and storage cost.

**Key words:** heterogeneous environments; heterogeneity; minimum redundancy; distributed; cloud storage

## 1 引言

云存储服务在过去几年中已经得到普及, 这些服务允许用户在异地存储数据. 为了存储来自数千用户的大量数据, 云存储系统在分布式存储基础上

构建服务, 比集中式解决方案更具可伸缩性和可靠性. 当前分布式数据存储主要采取两种方式: 网格方式和 P2P 覆盖方式. 为了实现所需的数据可用性, 所有这些数据中心, 需要对数据进行冗余存储<sup>[1]</sup>. 然而, 由于使用冗余增加了存储和通信成本, 因此优化

收稿日期: 2017-08-20; 修回日期: 2017-09-19

基金项目: 国家自然科学基金(61202006; 61472237); 南京大学计算机软件新技术国家重点实验室开放课题(KFKT2016B18)

这种冗余是保持系统可伸缩性的关键<sup>[2-3]</sup>. 通过减少冗余, 分布式存储系统可以减少存储和通信成本. 然而, 理论上存在限制, 即不增加存储成本就不能降低通信成本, 反之亦然. 传统上, 存储系统主要通过平衡对等体之间的成本<sup>[4]</sup>, 设计能够提供更好的通信存储, 使其逐渐逼近理论极限. 所有这些工作都在一个共同的约束下进行解决方案构建: 它们假定系统中所有存储节点具有均匀属性.

本文的主要目的是通过考虑云系统或其他现存分布式系统中存在的真正的非均质性来优化分布式存储系统. 更具体地说, 这里主要专注于通过考虑节点的在线/离线模式来减少冗余.

## 2 预备知识

### 2.1 数据存储模型

考虑稳定状态下的分布式存储系统, 即节点数和存储对象数恒定. 令  $H$  表示系统中所有存储节点集<sup>[5-6]</sup>. 然后, 存储过程将每个数据对象存储在一个小的节点子集  $N$  中,  $N \subset H$ . 不考虑  $N$  中节点如何选取. 一种可能的解决方案是向集中式目录服务请求一个具有空闲存储容量的节点列表. 但是, 为了简化计算, 采取方式是在  $H$  中采取随机选取的方式构建子集  $N$ .

此外, 设定  $A(t)$  和  $U(t)$  分别为子集  $N$  中在时刻  $t$  的可用节点和不可用节点. 显然,  $N = A(t) \cup U(t)$ ,  $\forall t \geq 0$ . 则可定义节点  $i$  的平均在线可用性:

$$a_i = \frac{1}{\tau_i} \int_{t_{0,i}}^{t_{0,i}+\tau_i} \Pr[i \in A(t)] dt \quad (1)$$

式中,  $\forall i \in N$ .  $t_{0,i}$  是节点  $i$  加入系统时间,  $\tau_i$  是节点  $i$  在系统中的生存周期. 假定对每个节点进行长时间的监测后, 可得到一个很好节点平均可用度估计. 然后, 可将把存储系统分成两个不同过程: 存储过程和恢复过程, 具体定义如下:

(1) 存储过程: 该过程利用  $\langle k, n \rangle$  可擦除代码来存储系统对象. 这意味着每个数据对象被分割成  $k$  个大小相等的块. 然后, 这些块被用来生成  $n$  个冗余块 ( $n \geq k$ ), 最后存储在一组存储节点  $n$  中. 该方案的冗余率  $r$  为:  $r = n/k$ . 为定义每个节点  $i$  所持有数据块数量, 存储过程使用赋值函数  $g, g: N \rightarrow \mathbb{N}$ , 其中  $g(i)$  表示每个节点  $i$  所赋予的冗余块数量. 这个函数必须确保  $\sum_{i \in N} g(i) = n$ , 且有  $g(i) \geq 1, \forall i \in N$ .

使用擦除代码, 存储过程需决定存储每个数据对象  $r$  所需冗余. 为此, 存储过程生成大的冗余块

“pool”, 满足  $n > |N|$ . 最后, 这些块进行分散, 并使用  $N$  中节点存储. 根据  $N$  的大小和  $n$  的取值, 存储过程可用三种不同的方式工作:

方式 1:  $n > |N|$ : 冗余块比存储节点多. 存储过程必须决定哪个节点存储更多的块.

方式 2:  $n = |N|$ : 数据块的数量与存储节点完全相同. 存储过程向每个节点存储一个数据块.

方式 3:  $n < |N|$ : 数据块比存储节点少. 因为有些节点不会存储任何块, 可以把这个案例看作是一个较小的存储节点集  $N'$ , 其中  $n = |N'|$ . 然后, 该方式退化为方式 2.

(2) 恢复过程: 该过程负责从  $n$  个冗余块中检索  $k$ , 并重建原始数据对象. 为保证恢复过程成功率, 存储系统必须确保至少有可用的  $k$  个冗余块存储在在线节点中, 其可称为数据的可用性. 令  $G(t) = \sum_{i \in A(t)} g(i)$  是一个离散的随机变量, 表示在时刻  $t$  存储在网络节点上的块数. 使用这个随机变量, 数据可用性  $d$  可以定义为:

$$d = \Pr[G(t) \geq k] \quad (2)$$

式中,  $\forall t \geq 0$ . 此数据可用性  $d$  表示在线检测至少  $k$  个冗余块的概率. 由于节点并不总是可用的, 所以不可能假定完美的数据可用性 ( $d = 1$ ). 存储过程的目的是保证数据可用性  $d$  高于期望的可用性  $\hat{d}$ , 满足  $d \geq \hat{d}$ .

### 2.2 问题描述

现有存储系统不考虑单个节点可用性  $a_i$ , 而是使用节点平均可用性作为其模型构建基础:

$$\bar{a} = \frac{1}{|N|} \sum_{i \in N} a_i \quad (3)$$

在不考虑非均质性情况下, 所有节点都被平等对待, 然后, 分配函数将所有相同数量的冗余块存储到所有节点<sup>[7]</sup>. 反之会导致选择节点存储产生过多冗余. 在均匀条件下, 存储系统假设  $n = |N|$ . 然后, 赋值函数为:  $g(i) = 1, \forall i \in N$ .

然而, 这种同质化策略导致存储系统产生不精确结果, 从而产生更多冗余. 这种冗余优化对任何旨在降低系统成本的存储系统至关重要, 例如带宽和存储开销成本等. 在本文中, 考虑分布式存储系统真实有效性, 利用异质性可降低预期可用性冗余量  $\hat{d}$ . 我们的方法分为三个主要步骤:

步骤 1: 提供一个分析框架来测量在异构场景中真实数据可用性  $d$ .

步骤 2: 给定一个固定的数据冗余, 设计一个最大化  $d$  的赋值函数  $g$ .

步骤3:找到保证数据可用性大于期望值的最小数据冗余  $d \geq \hat{d}$ .

### 3 测量数据的可用性

我们提供了特定存储场景中测量数据可用性  $d$  的基本框架.正如我们前面定义的,数据可用性是依赖于几个参数的概率度量.在所有这些参数中,对  $d$  的测量采用形式:(1)擦除码参数  $k$  和  $n$ ;(2)赋值函数  $g$ ;(3)节点集  $N$ .目标是获得函数  $D$  来测量数据可用性,  $d = D(k, n, g, N)$ .

#### 3.1 节点异构性假设

考虑节点  $N$  的幂集,  $2^N$ , 表示在线节点的所有可能组合的集合.令  $A, A \subset 2^N$ , 为其一种可能的组合.然后将组合  $A$  发生的事件表示为  $Q_A$ .由于节点的可用性是独立的,则有:

$$\Pr[Q_A] = \prod_{i \in A} a_i \prod_{i \in N \setminus A} (1 - a_i) \quad (4)$$

此外,令  $L_k, L_k \subset 2^N$ , 包含可用节点组合子集,它们一起存储  $k$  个不同冗余块.则可得:

$$L_k = \{A: A \in 2^N, \sum_{i \in A} g(i) \geq k\} \quad (5)$$

由于是在稳定状态下分析系统,所以我们随时都可引用在线数据块的数量,即  $G = \sum_{i \in A} g(i)$ .基于该公式,可将公式(2)改写为:

$$d = \Pr[G \geq k] = \sum_{A \in 2^N} \Pr[G \geq k | Q_A] \quad (6)$$

根据公式(5)中  $L_k$  的定义,则有:

$$\Pr[G \geq k | Q_A] = \begin{cases} \Pr[Q_A], & \text{if } A \in L_k \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

然后,根据公式(7)和公式(4),公式(6)可改写为如下形式:

$$d = \sum_{A \in L_k} \Pr[Q_A] = \sum_{A \in L_k} \left[ \prod_{i \in A} a_i \prod_{i \in N \setminus A} (1 - a_i) \right] \quad (8)$$

主要缺点是,如果  $N$  取值很大,则算法计算复杂度过大.同时,根据公式(5)可知,当  $k$  减小时,  $L_k$  中可用节点组合数增加.则对于式(8)  $d$  测量值计算,  $k=1$  是其最差情况.考虑  $g(i)=1, \forall i \in A$ , 且  $n = |N|$  的简单情形,则  $L_1 = 2^N \setminus \{\emptyset\}$ , 则可得  $|L_1| = 2^{|N|} - 1$ .这意味着使用异构通用表达的计算复杂度为  $O(2^n)$ .因此,对于规模较大的节点  $N$  集,利用公式(8)对测量  $d$  计算不可行.对于普通设备,当节点存储节点集大于20时,其计算时间大于1小时.

#### 3.2 节点可用性均匀假设

假设1:(均匀性假设)考虑同质节点可用性时,假定  $n = |N|$ , 且每个节点只存储一个数据块,即  $g(i)=1, \forall i \in N$ .考虑到同质性假设,我们可以重新定义公式(5)中  $L_k$ :

$$L_k = \{A: A \in 2^N, |A| \geq k\} \quad (9)$$

备注1:(平均节点可用性)对于同质化情形,  $a_i = \bar{a}, \forall i \in N$ , 其中  $\bar{a}$  为公式(3)定义的平均节点可用性.

在同质化假设下,数据可用性度量  $d$  的复杂性大大降低.则根据备注1可改写公式(4)为:

$$\begin{aligned} \Pr[Q_A] &= \prod_{i \in A} \bar{a} \prod_{i \in N \setminus A} (1 - \bar{a}) \\ &= \bar{a}^{|A|} (1 - \bar{a})^{|N \setminus A|} \end{aligned} \quad (10)$$

定义子集  $L'_x$  作为包含可用节点子集,可完全存储  $x$  块:  $L'_x = \{L: L \in L_k, |L| = x\}$ , 对于同质化情形,该子集满足  $|L'_x| = \binom{n}{x}$ .公式(8)改写:

$$d = \sum_{A \in L_k} \Pr[Q_A] = \sum_{x=k}^n \sum_{A \in L'_x} \Pr[Q_A] \quad (11)$$

由此可得如下定理:

**定理1:** 对于所有的  $\forall A_1, A_2 \in L'_x$  和  $x \in [1, n]$ , 存在  $\Pr[Q_{A_1}] = \Pr[Q_{A_2}]$ .

证明:该定理可结合公式(10)利用反证法进行验证,证明略.

利用定理1和  $|L'_x| = \binom{n}{x}$ , 可对公式(11)进行改写:

$$d = \sum_{x=k}^n \binom{n}{x} \Pr[Q_A] = \sum_{x=k}^n \binom{n}{x} \bar{a}^x (1 - \bar{a})^{n-x} \quad (12)$$

利用公式(12)进行数据可用性  $d$  的计算复杂度降为  $O(n)$ .

#### 3.3 聚类有效性

令  $C$  表示集合与质心的可用性,则对于每个质心可用性  $a \in C$ ,  $N_a$  代表  $a$  群集:

$$N_a = \{i: i \in N, a_i = a \pm \epsilon\} \quad (13)$$

式中,  $\epsilon$  定义集群内的最大可用性误差.则集群应满足:

$$N = \bigcup_{a \in C} N_a \quad (14)$$

假设2:(集群同质化)由于相同集群中的所有节点都被认为具有相同的可用性,所以其赋值函数将具有等价性.这意味着  $g(i_1) = g(i_2), \forall i_1, i_2$

$\in N_a$ .

为了表示集群中在线节点的所有可能组合,定义元组集合  $Z_a$ , 包含: (1) 集群中的节点总数; (2) 集群中的平均节点可用度; (3) 在线节点数. 具体形式为:

$$Z_a = \{(|N_a|, a, x)\}_{x=0}^{|N_a|} \quad (15)$$

假定集群中所有节点都有相同网络可用性, 集群中节点数二项式分布, 其概率为  $a$ , 大小为  $|N_a|$ . 对于  $z \in Z_a$ , 可对  $N_a$  中节点  $z_3$  概率进行测量  $f(z_3; z_1, z_2)$ . 由于节点可用性是具有独立性, 且集群网络节点数服从二项式分布, 则可测量  $Q_A$  概率为:

$$\Pr[Q_A] = \prod_{z \in A} f(z_3; z_1, z_2) \quad (16)$$

此外, 定义组合集  $z_k$ , 其中可用节点至少存储  $k$  个冗余数据块:

$$z_k = \{A: A \in \prod_{a \in C} Z_a, \sum_{z \in A} z_3 g(z_2) \geq k\} \quad (17)$$

由此可对公式(7)进行改写:

$$\Pr[G \geq k | Q_A] = \begin{cases} \Pr[Q_A], & \text{if } A \in z_k \\ 0, & \text{otherwise} \end{cases} \quad (18)$$

进而可对公式(6)可用性  $d$  计算进行改进:

$$\begin{aligned} d &= \sum_{A \in z_k} \left[ \prod_{z \in A} f(z_3; z_1, z_2) \right] \\ &= \sum_{A \in z_k} \left[ \prod_{z \in A} \begin{pmatrix} z_1 \\ z_3 \end{pmatrix} z_2^{z_3} (1 - z_2)^{z_1 - z_3} \right] \end{aligned} \quad (19)$$

其计算复杂度依赖于  $z_k$  大小.

#### 3.4 蒙特卡洛近似

随机生成集合  $S_\omega$ , 其从  $2^N$  中采集  $\omega$  个样本. 此集合包含  $x$  所有可能在线节点组合. 组合  $A \in S_\omega$  根据每个节点个体可用性  $a_i$  选择, 即  $\Pr[i \in A] = a_i$ . 则可得真实数据可用性  $d$  近似值:

$$d_\omega = \frac{|\{A: A \in S_\omega, \sum_{i \in A} g(i) \geq k\}|}{\omega} \quad (20)$$

随样本数增加, 其值将逐渐接近真实  $d$  值, 即  $d = \lim_{\omega \rightarrow \infty} d_\omega$ . 算法 1 给出其迭代计算过程.

算法 1: 数据可用性迭代近似

```

1. successes ← 0
2. iteration ← ω
3. while iteration > 0 do
4. block ← 0
5.   for i ∈ A do
6.     if rand() ≤ ai do
7. block ← block + g(i);
8.   endif

```

```

9.   endfor
10.  if block ≥ k then
11. successes ← successes + 1
12.  endif
13. iteration ← iteration + 1
14. endwhile
15. dω ← successes/iteration

```

因为  $d_\omega$  的计算时间与  $\omega$  呈正比. 则令  $\hat{\omega}$  表示可确保低可用误差  $\epsilon$  的最小  $\omega$  取值:

$$\hat{\omega}: = \min\{\omega: (d_\omega - d)^2 \leq \epsilon\} \quad (21)$$

为了查找  $\hat{\omega}$ , 初始设定  $\hat{\omega} = 1$ , 通过如算法 1 所示的迭代过程进行查找.

## 4 最优赋值函数

### 4.1 搜索空间定义

令  $\mathbb{R}^{|N|}$  是笛卡尔  $|N|$  空间, 即  $\mathbb{R}^{|N|}$  为欧氏  $|N|$  空间正交仿射框架:  $\mathbf{R} = \{O; \vec{e}_1, \vec{e}_2, \dots, \vec{e}_{|N|}\}$ . 然后, 假定  $N$  为有序集, 对应于数据帧  $\mathbf{R}$  上点  $p \in \mathbb{R}^{|N|}$  的所有节点赋值为  $[g(i)]_{i \in A}$ , 其笛卡尔坐标为  $(x_1, x_2, \dots, x_{|N|})$ . 每个组件  $x_i$  对应于  $N$  中分配给第  $i$  节点的冗余块的数量.

基于整个搜索空间  $\mathbb{R}^{|N|}$ , 只对满足赋值函数要求的可能解的一小部分感兴趣, 即  $g: \sum_{i \in N} g(i) = n$ . 这个要求限制了搜索空间  $S$  到超平面  $\pi_n$  的正区域:

$$\pi_n = \sum_{i=1}^{|N|} x_i = n \quad (22)$$

在进行最优赋值函数查找中, 可采用 PSO 算法 (PSO 算法不再赘述). 但是, 不能直接在  $S$  中随机设置粒子, 并令其自由移动. 如果这样处理, 则粒子群算法将随机更新每个粒子在所有维度中的位置和速度, 造成超范围移动. 为了解决这个缺点, 在  $\mathbb{R}^{|N|}$  参考帧内移动, 在平面  $\pi_n$  内设定  $|N| - 1$  向量. 图 1 给出帧移动的一个简单示例.

该示例表示三节点  $N = \{a, b, c\}$  节点集, 每个轴代表分配给这些节点的冗余块数量, 其表达了从帧  $(1, 3, 2) \in \mathbf{R}$  到帧  $(1, 3, 0) \in \mathbf{R}'$  的转换过程. 可观测到,  $\mathbf{R}'$  的第三维度始终为零. 此外, 因为处理数据块分配, 因此只能在  $\mathbf{R}'$  的整数位置使用.  $\mathbf{R} \rightarrow \mathbf{R}'$  转换过程中的一个特性是, 当  $\mathbf{R}'$  的整数位置转换回  $\mathbf{R}$  过程中, 它们在  $\mathbf{R}$  中也有整数坐标, 该特性允许粒子在  $\mathbf{R}'$  上自由移动. 定理 3 对该特性进行描述. 首先引入所需几何定义. 对于图 1 所示结构, 如果存在:

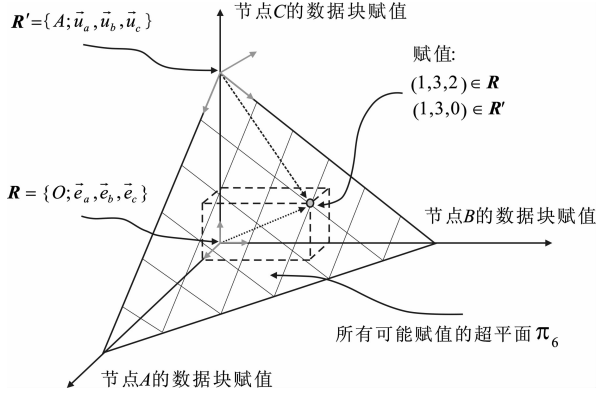


图 1 帧移动简单示例

$$u_i = \begin{cases} \vec{e}_i - \vec{e}_{|N|}, & \text{if } 1 \leq i \leq |N|-1 \\ \sum_{i=1}^{|N|} \vec{e}_i, & \text{if } i = |N| \end{cases} \quad (23)$$

可很容易地得到如下  $\pi_n$  的矢量方程:

$$\pi_n \equiv X = A + \sum_{i=1}^{|N|-1} \lambda_i u_i \quad (24)$$

式中,  $\lambda_i \in \mathbb{R}$ ,  $A$  为  $\pi_n$  和  $|N|^{\text{th}}$  的交集;  $A = \pi_n \cap r_{|N|}$ .  $\mathbb{R}$  上点  $A$  坐标是  $(0, 0, \dots, n)$ . 向量  $\vec{u}_{|N|}$  与  $\pi_n$  满足正交关系. 令  $(x'_1, x'_2, \dots, x'_{|N|})$  为点  $p \in \mathbb{R}^{|N|}$  新仿射结构  $L' = \{A; \vec{u}_1, \vec{u}_2, \dots, \vec{u}_{|N|}\}$  上坐标. 然后, 如果最近坐标  $x'_{|N|} = 0$ , 则  $p \in \pi_n$ .

**定理 2** 如果点  $p \in \pi_n$  在帧  $R'$  上具有整数坐标, 则点  $p \in \pi_n$  在帧  $R$  上也具有整数坐标, 即:

$$\begin{aligned} p &= (x_1, x_2, \dots, x_{|N|})_R \\ &= (x'_1, x'_2, \dots, x'_{|N|}, 0)_{R'} \end{aligned} \quad (25)$$

则可得:

$$\{x'_i\}_{i=1}^{|N|} \subset \mathbb{Z} \Rightarrow \{x_i\}_{i=1}^{|N|} \subset \mathbb{Z} \quad (26)$$

证明: 考虑两坐标系之间的代数关系对该定理进行证明:

$$M \begin{pmatrix} x'_1 \\ \vdots \\ x'_{|N|-1} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ n \end{pmatrix} = \begin{pmatrix} x_1 \\ \vdots \\ x_{|N|-1} \\ x_{|N|} \end{pmatrix} \quad (27)$$

式中, 左式第二项表示帧  $R$  和  $R'$  之间的坐标转换,  $M$  是旋转和缩放矩阵, 形式为:

$$M = \begin{pmatrix} 1 & 0 & 0 & \cdots & 1 \\ 0 & 1 & 0 & \cdots & 1 \\ 0 & 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 \end{pmatrix} \quad (28)$$

则可得:  $x_i = x'_i$ , 其中  $1 \leq i \leq |N|-1, x_{|N|} = -\sum_{i=1}^{|N|-1} x'_i + n$ . 定理 2 得证.

## 4.2 最佳冗余确定

一旦确定如何测量数据的可用性和最优分配函数, 就只能找到保证最小数据可用性  $\hat{d}$  所需的最小冗余量. 寻找冗余意味着寻找可用性  $\hat{d}$  所需的最小比例  $r = n/k$ . 但是, 因为涉及到两个参数  $k$  和  $n$ , 因此对于相同的  $\hat{d}$ , 存在的组合方式很多. 现有文献中处理方式, 是设定  $k$  的初始值, 然后令  $n$  从  $n = k$  开始, 迭代查找所需的可用性  $\hat{d}$  取值, 这种算法得到的并非最佳. 本文中采用的方式是设定  $n$  的初始值为  $n = |N| \cdot \beta$ , 对于较大的  $\beta$  取值, 令  $k$  从  $k = n$  开始降低, 直到得到理想的  $\hat{d}$  取值. 这样可得到比  $N$  中节点更多的冗余块, 并且可很容易地存储更多的块到更稳定的节点. 根据上述描述有:

$$k := \max\{k: D(k, n, g, N) \geq \hat{d}, k \in [1, n]\} \quad (29)$$

**定理 3** 为获得  $k$ , 存在如下关系:  $D(k, n, g, N) \geq D(k-1, n, g, N), \forall k \in [1, n]$ .

证明: 从一般的数据可用性的表达式 (公式 (8)), 可很容易地推断函数  $D$ , 或其表达  $d$ , 随着  $L_k$  的大小增加而单调增加. 根据公式 (5), 可很容易得到  $|L_k| \leq |L_{k-1}|$ , 由此可得定理 3 成立.

根据定理 3 可采用简单的一维迭代形式进行最佳冗余确定, 算法过程略.

## 5 实验分析

### 5.1 最优分配实验

为了找到最佳的分配功能, 研究几个不同大小、不同节点可用性的不同运行场景下的 PSO 优化结果. 将粒子群算法的惯性参数设置为  $\omega_{\min} = 0.5$ 、 $\omega_{\max} = 0.75$  和  $i_{\max} = 50$ , 常量  $\xi_1, \xi_2 = 1$ . 粒子种群规模设置为 100.

由于分布式存储系统中缺乏真正的可用性的轨迹, 这里利用来自不同  $\beta$  分布的随机数据进行表示, 其均值为  $\{0.25, 0.5, 0.75\}$ , 方差为  $\{0.01, 0.02, 0.03, 0.04\}$ . 为了显示  $k$  值对算法的影响, 这里测试四组不同的  $k$  值,  $k \in \{n/5, n/3, n/2, n/1.5\}$ , 其中  $n = 10|N|$ .

图 2 所示为  $|N| = 10$  时, 对于不同  $k$  值的实验结果.

根据图 2 所示结果可知, 最优分配倾向于在不丢弃低可用性的情况下向稳定节点分配更多冗余数

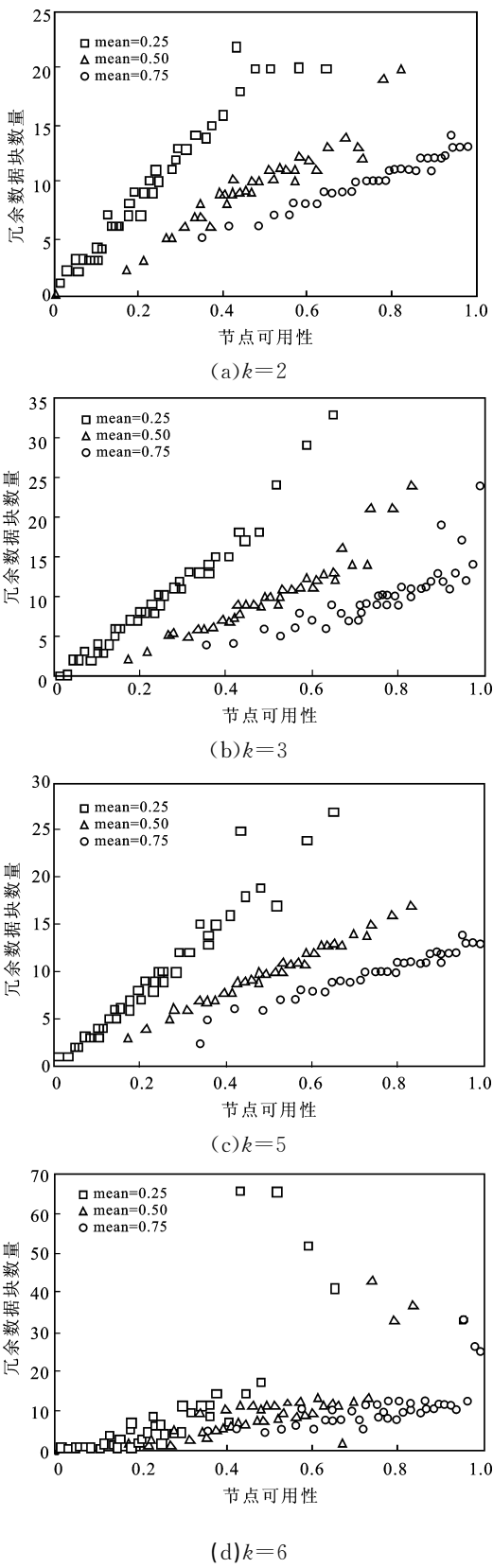


图 2 较小存储节点集情形

据. 此外, 这个赋值也趋向于沿着一条通过坐标原点的直线对齐. 图中的误差偏离点主要是因为选用非确定 PSO 算法的原因.

5.2 冗余实验分析

这里定义冗余率度量指标(RSR), 这个度量指标可表征考虑异构存储系统而不是同质存储系统而导致的冗余节省. 令  $r_{\text{homo}}$  和  $r_{\text{heter}}$  分别为同质系统和异构系统, 在保证数据可用性  $\hat{d}$  的数据冗余. RSR 指标可定义为:

$$\text{RSR} = \left(1 - \frac{r_{\text{heter}}}{r_{\text{homo}}}\right) \times 100 \tag{30}$$

同样针对上述设定的  $|N| = 10$  和  $|N| = 100$  两种情形. 对比算法选取文献[8] 基于块的云存储服务共享方案、文献[9] 基于公开验证性和可用性的云存储技术和文献[10] 基于可搜索数据文件共享的云存储方案. 利用上述参数设置, 分两种情况进行算法运行研究: (1) 较小存储节点集情况  $|N| = 10$ , 利用公式(8) 所示的适应度函数一般解析式; (2) 利用较大的节点集  $|N| = 100$ , 利用蒙特卡洛法获得  $d$  近似. 实验对比结果见表 1 所示.

| 表 1 冗余实验对比        |       |       |        | %     |
|-------------------|-------|-------|--------|-------|
| 参数                | 文献[8] | 文献[9] | 文献[10] | 本文算法  |
| $ N  = 10$        |       |       |        |       |
| $\hat{d} = 0.9$   | 42.23 | 16.52 | 14.28  | 11.64 |
| $\hat{d} = 0.99$  | 45.63 | 19.52 | 16.74  | 12.81 |
| $\hat{d} = 0.999$ | 52.19 | 25.67 | 22.19  | 15.43 |
| $ N  = 100$       |       |       |        |       |
| $\hat{d} = 0.9$   | 32.71 | 23.41 | 21.42  | 18.59 |
| $\hat{d} = 0.99$  | 35.42 | 25.91 | 24.67  | 20.64 |
| $\hat{d} = 0.999$ | 38.42 | 28.76 | 26.82  | 22.16 |

根据表 1 数据可知, 在同等数据可用性指标下, 本文算法的数据冗余度最低, 这体现了本文算法在计算效率上的优势. 同时可看出, 随着网络规模的增加, 数据的冗余度降低, 而随着可用性指标  $\hat{d}$  的增加, 所需的数据的冗余度也增加.

6 结束语

现有的云存储服务的设计多是建立在存储后端构成的分布式资源同质基础上, 这种假设考虑到云系统中所有节点存在唯一在线可用性, 然后根据这个假设来优化它们的数据冗余. 然而, 这种假设简化了数据可用性测量, 但它引入了一个导致数据冗余增加的误差, 会降低数据计算效率. 在本文中, 我们研究了异构云基础设施对异构节点可用性影响性能. 为此, 我们提出了一个分析框架, 主要涉及三个方面: (1) 异构存储基础设施中数据可用性的度量算 (下转第 34 页)

- [6] Ke Z, Ai H P. Optimization Of Ceph reads/writes based on multi-threaded algorithms[C]//IEEE International Conference on High Performance Computing and Communications; IEEE International Conference on Smart City; IEEE International Conference on Data Science & Systems, New York, 2016:719-725.
- [7] 刘飞,蒋德钧,张欢,等. 异构存储感知的 Ceph 存储系统数据放置方法[J]. 计算机科学, 2017, 44(6):17-22.
- [8] 陈凌剑,王勇,俸皓. 基于网络延时的 Ceph 存储性能优化方法[J]. 微电子学与计算机, 2017, 34(6):84-88.
- [9] Mackey G, Sehrish S, Wang J. Improving metadata management for small files in HDFS[C]//IEEE Inter-

national Conference on Cluster Computing and Workshops. New Orleans, LA, USA. 2009:1-4.

## 作者简介:

**刘辉勇** 男, (1992-), 硕士研究生. 研究方向为云存储.

E-mail: 1126610758@qq. com.

**王 勇** 男, (1964-), 博士, 教授. 研究方向为云计算、计算机网络技术及应用、信息安全等.

**俸 皓** 男, (1978-), 博士研究生. 研究方向为计算机网络、无线传感器网络、软件定义网络及云计算等.

(上接第 26 页)

法; (2) 为存储节点设置冗余块的最佳方法; (3) 确定最低数据冗余以达到所需服务质量的机制. 由于经济收入是云存储服务的一个关键因素, 下一步工作中, 我们将对工作进行扩展, 以便在今后的工作中进行考虑.

## 参考文献:

- [1] Chen Minghung, Tung Yuchih, Hung Shihhao, et al. Availability is not enough: minimizing joint response time in peer-assisted cloud storage systems[J]. IEEE Systems Journal, 2016, 10(4): 1424-1434.
- [2] Yu Jia, Wang Huaqun. Strong key-exposure resilient auditing for secure cloud storage[J]. IEEE Transactions on Information Forensics and Security, 2017, 12(8): 1931-1940.
- [3] 刘伯红, 史慧丽. 一种高效属性撤销的云存储访问控制方案[J]. 微电子学与计算机, 2016, 33(4): 41-45.
- [4] Yan Hao, Li Jiguo, Han Jinguang, et al. A novel efficient remote data possession checking protocol in cloud storage[J]. IEEE Transactions on Information Forensics and Security, 2017, 12(1): 78-88.
- [5] Yu Jia, Ren Kui, Wang Cong. Enabling cloud storage auditing with verifiable outsourcing of key updates[J]. IEEE Transactions on Information Forensics and Secu-

ity, 2016, 11(6): 1362-1375.

- [6] 刘绍刚. 基于 NAS 的私有云存储平台的设计与实现[J]. 计算机测量与控制, 2017, 25(4): 205-208.
- [7] 谭振华, 杨广明, 王兴伟, 等. 面向云存储的多维球面门限秘密共享方案[J]. 软件学报, 2016, 27(11): 2912-2928.
- [8] Abdul Nasir Khan, Mat Kiah M L, Mazhar Ali, et al. BSS: block - based sharing scheme for secure data storage services in mobile cloud environment[J]. The Journal of Supercomputing, 2014, 70(2): 946-976.
- [9] Rashmi M Jogdand, Goudar R H, Gazal Begum Sayed, et al. Enabling public verifiability and availability for secure data storage in cloud computing[J]. Evolving Systems, 2015, 6(1): 55-65.
- [10] Pitchai R, Jayashri S, Raja J. Searchable encrypted data file sharing method using public cloud service for secure storage in cloud computing[J]. Wireless Personal Communications, 2016, 90(2): 947-960.

## 作者简介:

**胡力文** 女, (1983-), 硕士研究生, 讲师. 研究方向为计算机软件技术理论. E-mail: 61661605@qq. com.

**李启明** 男, (1982-), 博士, 副教授. 研究方向为大数据分析、高性能计算.